L2/18-203

Re: Coded Hashes of Arbitrary Images (L2/16-105)

From: Mark Davis

Date: 2018-05-22

The document “Coded Hashes of Arbitrary Images” (L2/16-105) presents a proposal for an extensible mechanism for emoji (and other images) encoded in Unicode. The proposal was not accepted by the UTC as it stood, nor did the authors follow up with revisions, but it isn’t clear to many people what the issues are. Why was this seemingly attractive proposal not accepted? Since this topic comes up, this my recollection of the issues with that proposal.

The basic idea of the proposal is that you'd have a sequence of characters representing a hash of an image and could interchange that hash. That would allow anyone to come up with their own emoji and exchange them.

However, there was never a complete story on how this could work in practice. In particular, suppose that a program on a mobile phone gets one of these sequences in an email or other document or text. What does it do with it? Where does it get the image from? Is there a registry somewhere? How does that function? Who maintains it? And so on.

There are definitely security concerns: People remember the problems with HTML email containing hidden, invisible 1x1 pixel images that accessed a server, letting that the owners of that server know the IP addresses of people were viewing the message. Having that happen with plain text would not be acceptable.

The UTC had considered other alternatives even before that paper was submitted, such as using the TAG mechanism, but with namespaces using URLs. There is an older document at L2/16-008, and even older discussions that predated that document. The idea (modified to bring it up to date) would be an emoji TAG sequence like:

X<u>ABC.CO/qybUFnY7Y8w</u>✦

In this, X is a fallback emoji, and underline means TAG characters, as per valid-emoji-tag-sequences.

That way the implementation would know to go to http://abc.co/qybUFnY7Y8w, which could resolve to https://www.youtube.com/watch?v=qybUFnY7Y8w (thus allowing not only static but also animated images...). There too, however, security issues present themselves.

One could also use a dataURL approach, with the following and resolving to ...

X<u>XpngiVBORw0KGgoAAAANSUh...</u>✦

However, such images are huge, and would definitely stress the notion of plain text, and cost a lot in bandwidth.

It is unlikely that “Coded Hashes of Arbitrary Images” (L2/16-105) would be pursued until and if a substantially better (aka feasible) revision proposal is made.